2017 46th International Conference on Parallel Processing

CELIA: Cost-time Performance of Elastic Applications on Cloud

Sunimal Rathnayake, Dumitrel Loghin, Yong Meng Teo
Department of Computer Science
National University of Singapore
{*sunimalr, dumitrel, teoym*}*@comp.nus.edu.sg*

***Abstract*— Clouds offer great flexibility for scaling applications due to the wide spectrum of resources with different cost-performance, inherent resource elasticity and pay-per-use charging. However, determining cost-time-efficient cloud configurations to execute a given application in the large resource configuration space remains a key challenge. The growing importance of elastic applications for which the accuracy is a function of resource consumption introduces new opportunities to exploit resource elasticity on clouds. In this paper, we introduce CELIA, a measurement-driven analytical modeling approach to determine cost-time-optimal cloud resource configurations to execute a given elastic application with a time deadline and a cost budget. We evaluate CELIA with three representative elastic applications on more than ten million configurations consisting of Amazon EC2 resource types with different cost-performance. Using CELIA, we show that multiple cost-time Pareto-optimal configurations exist among feasible cloud configurations that execute an elastic application within a time deadline and cost budget. These Pareto-optimal configurations exhibit up to 30% cost savings for an elastic application representing n-body simulation. We investigate the impact of fixed-time scaling on the cost of executing elastic applications on cloud. We show that cost gradient with respect to resource demand is smaller when cloud resources with better cost-performance are used. Furthermore, we show that the relative increase in cost is always smaller compared to the relative reduction of execution time deadline. For example, tightening the execution time deadline by two-thirds incurs only 40% increase in cost for the n-body simulation application.**

***Keywords*-elastic application; cloud; cost-time performance; Pareto-optimal configuration**

I. Introduction

Cloud computing has advanced rapidly during the past decade and has attracted cloud providers such as Amazon and Google, among others. Cloud providers offer resource elasticity and a wide range of cloud resource types [3]. These resource types are characterized based on processor, memory and storage performance, among others. On the other hand, companies ranging from startups to large enterprises, and researchers in academia are adopting cloud services due to their flexible and low cost resource provisioning with pay-as-you-go pricing [17], [22]. However, choosing the right configuration to execute an application within a time deadline and a cost budget is a daunting task due to the large resource pool offered by the cloud.

In contrast to the widely known resource elasticity of clouds, the less explored application elasticity refers to scaling application resource demands to obtain better results. While in other works the term "elastic applications" refers to applications that scale their resource allocation based on changing resource demand [16], in this paper we refer to applications that produce results of different accuracy depending on the resource consumption. Examples of elastic applications include video and image processing, scientific simulation and bioinformatics. The elasticity of cloud resources has the potential to achieve scalable performance for elastic applications with a fixed-time deadline and a cost budget. Furthermore, this introduces an opportunity to trade-off the cost of execution for accuracy or quality of the results produced by the application. Thus, it is worthwhile to investigate the implications of executing elastic applications with a given time deadline and cost budget on cloud.

Amdhal's law [4] shows that the parallel speedup of an application is limited by its sequential fraction when the size of the application is fixed. We investigate the fixed-time scaling of applications when the application grows with the size of the parallel platform. This fixed-time scaling model in the context of elastic applications leads to two cases, (i) fixing both time deadline and accuracy while scaling the problem size or (ii) fixing both time deadline and problem size while scaling the accuracy to improve the quality of application output. Nonetheless, there are cases where problem size is also related with the accuracy of results, such as for n-body simulations. We investigate the effect of these two types of fixed-time scaling on cost with representative elastic applications.

Addressing these issues, we propose CELIA, a measurement-driven analytical modeling approach to determine cost-time optimal cloud configurations for executing a given elastic application with a time deadline and a cost budget. To match application resource demand with cloud resource capacity, we use the number of instructions executed as the proxy for resource demand and the instruction execution rate as the proxy for resource capacity. To measure application resource demand and cloud resource capacity, we conduct baseline executions of elastic applications on a local server and on selected cloud resources. Due to the virtualization of cloud resources, traditional performance monitoring techniques such as using CPU performance counters cannot be applied. Thus, we are constrained to use a local server with the same micro-architecture as target cloud resources to measure

2332-5690/17 $31.00 © 2017 IEEE
DOI 10.1109/ICPP.2017.43

IEEE computer society

instruction count. Moreover, as each elastic application has multiple application parameters and different execution profile, we are required to establish the relationship between application parameters and application resource demand.

Leveraging cloud elasticity for cost optimization has long been a hot research topic. We broadly classify related work into two categories, (i) resource elasticity where cost optimization is achieved through scaling cloud resources based on the resource demand of the application [19], [18], [13], [24], [15], [28], [7], [6] and, (ii) application elasticity where cost optimization is achieved by leveraging the application's ability to scale accuracy or quality [5], [10], [9]. Studies on cost optimizations based on resource elasticity include techniques such as autoscaling and resource migration, scheduling, checkpointing and model-based predictions. On the other hand, application elasticity has not been extensively studied. Existing work include trading-off quality of on-demand video applications for cost and frameworks to develop elastic algorithms that produce approximate results depending on the monetary investment.

In the context of exploring application elasticity on cloud, this paper makes the following key contributions:

1) We introduce CELIA, a measurement-driven analytical modeling approach to determine cost-time optimal cloud configurations that execute a given elastic application with a time deadline and cost budget. We evaluate CELIA on more than ten million configurations exposed by nine Amazon EC2 resource types, using three representative elastic applications covering domains such as video compression (x264), n-body simulations (galaxy) and bioinformatics (sand).
2) Using CELIA, we show that among feasible configurations that execute an elastic application within a time deadline and cost budget, the cloud exposes multiple cost-time Pareto-optimal configurations that minimize time and cost. We show that selecting a Pareto-optimal configuration among feasible configurations can save up to 30% of the cost for galaxy application.
3) We investigate the cost of fixed-time scaling of elastic applications' resource demand and observe that cost grows faster than resource demand when multiple cloud resources with different cost efficiency are used.
4) We show that the relative increase in cost when tightening the execution time deadline is smaller than the relative decrease of time deadline. For example, if the time deadline is tightened by two-thirds, the cost increases with just 40% for galaxy application.

The paper is organized as follows. Section II presents related work, and is followed by Section III where we describe CELIA, our modeling-based approach. Section IV presents the evaluation, including elastic applications characterization, cloud resources characterization, validation of our approach, and model-based analysis. Section V concludes the paper.

II. Related Work

We discuss related work on cost-time performance of the cloud under two main categories, (i) resource elasticity and (ii) application elasticity, and compare with our approach.

A. Resource Elasticity

Optimizing time and cost of executing cloud applications is a hot research topic. Many researchers have attempted to leverage pay-as-you-go pricing strategy and resource elasticity offered by clouds using various approaches.

Scalability is a key cloud characteristic that allows resource configuration to be changed dynamically according to the needs of the application being executed. This is achieved through the use of commercially available APIs such as AWS Autoscaling [2]. Mao et al. [19], [18] proposes leveraging autoscaling for cost optimization for cloud workflows using a performance model to estimate workload demand and resource capacity in terms of number of jobs. This approach relies on the user to estimate job processing times on each type of cloud resource. In contrast, our approach focuses on applications with a fixed time deadline and a cost budget and to predict execution time and cost, we use a combination of baseline measurements on cloud instances and a local server. Kokkinos et al. [13] and Sharma et al. [24] formulate cost optimization as an integer linear programming problem and determine a suitable cloud configuration to migrate an already running application. They utilize performance data retrieved for the current cloud configuration as inputs to the integer linear programming algorithm. Our analytical modeling approach is complementary to autoscaling and resource migration solutions as it could be used to determine the cost-time optimal cloud configuration to execute the application.

Zhang et al. [15], Kllapi et al. [12] and Zhou et al. [28] propose scheduling frameworks that optimize time and cost of execution for cloud workflows consisting of inter-dependent tasks. Similar to our work, Zhang's approach considers multiple cloud resource types and selects a combination of resource types as the cloud configuration.

Scheduling execution based on resource pricing is another optimization strategy applied in clouds that support spot pricing such as Amazon EC2. Marathe et al. [20] present an algorithm to find the best checkpointing strategy for executing an application on spot resources such that time and cost of execution are optimized by making use of historical spot pricing data. Due to price fluctuations, executing application on spot resources risks abrupt termination, thus, is difficult to guarantee time deadline satisfaction. Fault-tolerant mechanisms could be employed to eliminate termination risks. Gong et al. [7] address this issue by replicating execution on on-demand resources in addition to spot resources to guarantee time deadline satisfaction. However, this approach may exceed cost budget constraints. While these works consider spot instances, our paper focuses on on-demand resources.

Closer to our approach, Huang et al. [11] build an analytical model to predict the performance of resources and to map resources with workload demand estimated using TAU profiler [25]. Since resource demand estimation is solely based on the profiling on cloud by TAU profiler, the usage of their approach is limited on commercial cloud platforms due to the restrictions imposed due to virtualization. In comparison, our measurement-driven analytical modeling approach uses a combination of time measurements on selected cloud resources and performance counter readings using Linux `perf` utility on a local server to estimate application demand and cloud resource capacity, thus is applicable on commercial clouds. Model-driven resource allocation solution by Bicer et al. [6] divides the workload into equal sized chunks of data or jobs and predicts the time to execute data-intensive jobs concurrently on hybrid cloud environments. A limitation of this method is that when application resource demand is not a uniform function of time, workload execution time estimation would be inaccurate. In contrast, we first profile the application to determine its resource demand function with respect to problem size and accuracy, and thus, we provide a better estimation of resource demand, execution time and cost.

B. Application Elasticity

The concept of accuracy or quality of results in cloud computing has always been interpreted as the Quality-of-Service (QoS). QoS refers to the levels of performance, reliability and availability of cloud services and serves as a business metric within cloud stakeholders. QoS standards are enforced using service level agreements (SLA) that specify economic penalties for QoS violations. Thus, the trade-off between QoS and operational cost has been explored [5]. As QoS usually does not represent the quality or accuracy of the output produced by a cloud application, it is desirable to have a metric that quantifies the quality or accuracy of application output in consumer's terms and that allows consumer to set execution targets to achieve a desired quality of results. He et al. [10] work on Cost-Quality-of-Experience (QoE) trade-off is conceptually close to our idea of trading-off the accuracy of an application for cost savings. They build a theoretical model to exploit the cost-QoE trade-off for cloud based video content streaming providers. They characterize cloud resources in terms of video streaming performance and cost, and provide guidance for procuring the best resource type to satisfy consumers' desired QoE. While this approaches the problem from a cloud provider's perspective, our work approaches the problem from a consumer's perspective where the consumer executes an application with time deadline and cost budget constraints.

Han [9] and Guo et al. [8] investigate a class of algorithms they call elastic algorithms where iterative algorithms produce approximate results with monotonic accuracy at each step. These algorithms produce results of different quality depending on how long they execute. They propose to exploit this property to tradeoff cost of execution for compromises in accuracy. Also, they propose a framework to transform existing traditional algorithms into elastic algorithms. In comparison, our approach focuses on applications with implicit elasticity and relies on determining the relationship between the resource demand and application parameters in order to determine the optimal cloud resource configurations.

Table I
MODEL NOTATIONS

Symbol	Description
Elastic Applications	
$P_{n,a}$	an elastic application with problem size n and accuracy a
$P_{n',a'}$	a scale-down application with problem size n' and accuracy a'
$D_{Pn,a}$	resource demand of elastic application $P_{n,a}$
Cloud Resources	
I	set of all resource types
M	total number of resource types in I
$m_{i,max}$	maximum number of nodes available from resource type i
c_i	cost per unit time for resource type i
v_i	number of virtual processors in resource type i
$W_{i,vCPU}$	instruction execution rate of resource type i per virtual CPU
W_i	resource capacity of cloud resource type i
S	total number of cloud configurations
G	set of all cloud configurations
G_j	a cloud configuration j consisting of one or more resources
U_j	total resource capacity of configuration j
$C_{j,u}$	total cost per unit time of configuration j
Time Model	
T'	time deadline to execute $P_{n,a}$
T	predicted execution time for $P_{n,a}$
Cost Model	
C'	cost budget to execute $P_{n,a}$
C	predicted execution cost for $P_{n,a}$

III. APPROACH

In this section, we present CELIA, our measurement-driven analytical modeling approach to determine cost-time optimal cloud configurations.

A. Overview

Given an elastic application with a time deadline and a cost budget, and a set of cloud resources, CELIA determines cost-time Pareto-optimal cloud configurations that satisfy deadline and budget constraints, as illustrated in Figure 1. CELIA consists of three parts. Firstly, baseline runs of a scale-down versions of the elastic application are executed on both a local server and on cloud-based nodes to characterize application resource demand and cloud resource capacity. Secondly, our time and cost models take as input application and cloud resources characterizations to determine the time and cost of executing the elastic application on a set of cloud resources. Thirdly, our configuration selection algorithm finds cloud configurations that satisfy the time deadline and cost budget using our time and cost models. These configurations are filtered to determine Pareto-optimal configurations that minimize time and cost.

We explain in detail all three parts of CELIA using the notations in Table I. For each elastic application, we determine its resource demand by conducting baseline executions of different scale-down values of problem size, n', and accuracy, a', and by measuring the instruction count on a local server that has the same architecture as cloud resources. This instruction count is measured using non-intrusive hardware counters available on most modern processors. However, due to virtualization and security issues, cloud providers do not allow the usage of hardware counters and, thus, we are constrained to use a local server for these measurements. To estimate cloud resources capacities, we run the same scale-down versions of the application on each cloud instance type and measure the execution time. We compute the resource capacity of a cloud resource in terms of instructions executed per unit of time as the ratio between measured instruction count and measured execution time. This rate includes virtualization overhead imposed by the cloud platform, thus, we do not need to take it into account separately.

The number of instructions executed is used as a proxy to match resource demand of the application with the capacity of cloud resources. This matching enables us to determine cloud configurations capable of executing the given application with time deadline and cost budget constraints. Among these configurations, we select cost-time Pareto-optimal configurations that either minimize cost for a given deadline or minimize time for a given budget. The set of Pareto-optimal configurations represents the Pareto frontier in cost-time space. In Section IV, we study the impact of scaling problem size and accuracy on the Pareto-optimal configurations.

A cloud configuration is a combination of nodes from one or more types of cloud resources. For example, if there are three different resource types, the tuple <2,3,5> represents a configuration that uses two nodes from first resource type, three nodes from the second resource type and five nodes from the third resource type. Generalizing for M cloud resource types, a configuration G_j is a tuple $<m_{j,1}, m_{j,2}, ..., m_{j,M}>$, where $m_{j,i}$ is the number of nodes of type $i \in [1, M]$. This number of nodes can take integer values between 0 and $m_{i,max}$, hence, the total number of configurations exposed by cloud resources is

$$S = (\prod_{i=1}^{M}(m_{i,max} + 1)) - 1 \quad (1)$$

considering that the configuration with no nodes is not useful.

B. Time Model

Given an elastic application $P_{n,a}$ with problem size n and accuracy a, $D_{P_{n,a}}$ denotes its resource demand in terms of total number of instructions as a function of n and a. The time to execute this application on a cloud configuration $G_j =< m_{j,1}, m_{j,2}, ..., m_{j,M} >$ is

$$T = \frac{D_{P_{n,a}}}{U_j} \quad (2)$$

where U_j is the total capacity of configuration G_j expressed in instructions per second. We characterize both the resource demand of real elastic applications and the resource capacity of real cloud instances in Section IV.

The total resource capacity of configuration G_j is the summation of resource capacities of all nodes in this configuration,

$$U_j = \sum_{i=1}^{M}(m_{j,i} \times W_i) \quad (3)$$

We derive the computational capacity of a cloud resource using the instruction execution performance (MIPS) per virtual processor, $W_{i,vCPU}$, as derived from the baseline measurements on both local and cloud-based nodes. We assume that all cloud resources are multicore systems and that a virtual processor core in a cloud resource, *vCPU*, is a hyper-thread of the underlying physical processor. Thus, the resource capacity of cloud resource i comprised of v_i virtual processing cores is

$$W_i = W_{i,vCPU} \times v_i \quad (4)$$

In this paper, we focus on highly-parallelizable, compute-intensive applications where communication overhead is negligible, thus, we do not model communication overhead.

C. Cost Model

We derive the execution cost of an elastic application $P_{n,a}$ on a cloud configuration G_j by multiplying the execution

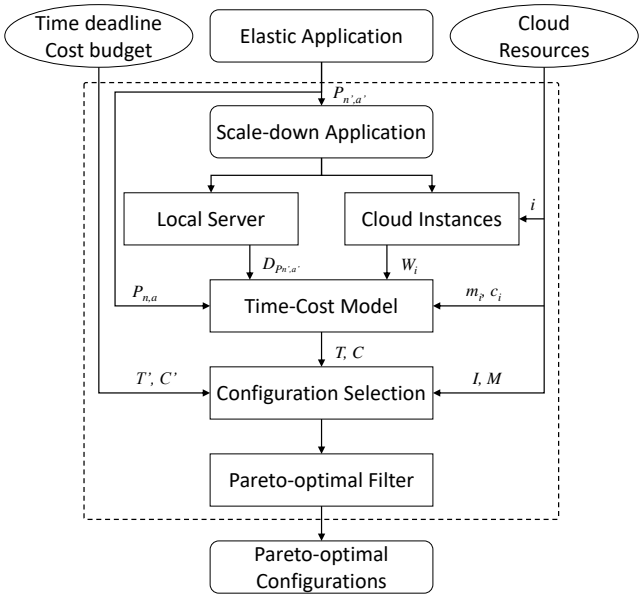


Figure 1. Overview of CELIA

Algorithm 1 Resource Configuration Selection

```
compute G using I and m_{i,max}
for each G_j in G do
    compute U_j, C_{j,u}, T, C
    if T < T' and C < C' then
        output G_j, T, C
    end if
end for
```

time T by the total cost per unit time of configuration's resources,

$$C = T \times C_{j,u} \tag{5}$$

where

$$C_{j,u} = \sum_{i=1}^{M} (m_{j,i} \times c_i) \tag{6}$$

Cloud pricing data is taken from the cloud vendor website.

D. Configuration Selection Algorithm

Our configuration selection algorithm generates a list of configurations that meet the execution time and budget constraints. As shown in the pseudocode listed in Algorithm 1, the execution time and cost for every configuration G_j are computed using our models. The predicted values are compared against deadline and budget constraints. Finally, the configuration list is passed through a Pareto-optimal filter [27] to get the optimal configurations, as shown in Figure 1. As our approach explores the entire configuration space, it guarantees to find all optimal configurations for a given resource demand with a time deadline and a cost budget.

IV. Evaluation

Evaluation is divided into five main sections. Firstly, elastic applications with different execution profiles are characterized to determine the relationship between application parameters and application resource demand. Secondly, cloud resources with different cost-performance profiles are characterized to estimate their computation capacities. Thirdly, we present a method to optimize cloud resource characterization. Fourthly, our cost-time model is validated against cloud measurements. Finally, we present an analysis of the results from our model-driven approach for the selected elastic applications on target Amazon EC2 resources.

A. Elastic Applications

To investigate the implications of application elasticity on cost-time performance of cloud configurations, we selected three elastic applications with different relationships between application parameters and application resource demand. Moreover, these applications are representing domains such as multimedia, astrophysics and bioinformatics, as summarized in Table II.

x264 [1] is a commercial video encoding application that takes a number of video clips, n, and distributes them among a number of independent processes running on one or multiple cluster nodes. The video encoding is done with an accuracy represented by the compression factor, f, that varies from 1 to 51. For simplicity, we fix the size of a single video clip to 75MB in our experiments. *Galaxy* [14] implements the n-body simulation of masses in a galaxy where input parameters are the number of masses, n, and the number of simulation steps s. Masses are distributed among MPI processes to compute the new positions for each mass at each simulation step. Simulation accuracy improves with increasing number of steps, thus, we use s as a proxy for accuracy. There are no theoretical upper bounds for n and s. *Sand* [21] is an application for genome sequence assembly. It aligns compatible genome sequences from a list of candidate sequences of size n. The quality threshold, t, specifies the degree of similarity for two candidate sequences to be aligned together. This application is implemented using a master-slave approach built on Work Queue platform [23]. The master process takes the list of candidate sequences, creates a set of alignment tasks and distributes them among slave processes. There is no upper bound for n and the meaningful range for t is from 0 to 1.

To determine the relationship between application parameters, such as problem size and accuracy, and application resource demand, we run each application with different problem sizes and accuracy levels on a local Intel Xeon E5-2630 v4 server and measure the instruction count using Linux `perf` utility. For x264, we range the problem size from 2 to 32 and f from 10 to 50. For galaxy, n ranges from 8,192 to 65,536 bodies and s ranges from 1,000 to 8,000 steps. For sand, n range is from 1 million to 64 million sequences, and t range is from 0.01 to 1.

As shown in Figure 2, x264 exhibits a linear relationship between n and resource demand, while the relationship between f and resource demand is quadratic. In galaxy, n is quadratically related to resource demand, while s affects resource demand in a linear way. In sand, resource demand grows linearly with n and logarithmically with t. These results show that selected elastic applications exhibit linear, quadratic and logarithmic resource demands as function of problem size and accuracy.

B. Cloud Resources

This section presents the characterization of selected cloud resources from Amazon EC2. To substantiate the application of our approach on the cloud, we selected nine Amazon EC2 resource types and up to five instances from each type, representing a large configuration space of 10,077,695 configurations. These resources are possessing different cost-performance, as shown in Table III. Amazon cloud resources are classified based on their performance characteristics into different categories such as compute-

Table II
ELASTIC APPLICATIONS

Application	Domain	Input	Problem Size
x264	video compression	number of videos (n), size of video, compression rate (f)	(n, f)
n-body galaxy simulation (*galaxy*)	astrophysics	number of masses (n), simulation steps (s)	(n, s)
SAND genome alignment (*sand*)	bioinformatics	genome sequences (n), threshold (t)	(n, t)

Table III
AMAZON EC2 CLOUD RESOURCE TYPES

Type	vCPUs	Frequency (GHz)	Memory (GB)	Storage[1] (GB)	Cost ($)
c4.large	2	2.9	3.75	EBS	0.105
c4.xlarge	4	2.9	7.5	EBS	0.209
c4.2xlarge	8	2.9	15	EBS	0.419
m4.large	2	2.3	8	EBS	0.133
m4.xlarge	4	2.3	16	EBS	0.266
m4.2xlarge	8	2.3	32	EBS	0.532
r3.large	2	2.5	15	32	0.166
r3.xlarge	4	2.5	30.5	80	0.333
r3.2xlarge	8	2.5	61	160	0.664

[1] EBS = Amazon Elastic Block Storage

intensive, general-purpose and memory-optimized, among others. Each of these categories consist of multiple resource types such as *large*, *xlarge* and *2xlarge*, labeled based on the number of vCPUs each resource type possesses. We select three resource categories representing compute-intensive *c4*, general-purpose *m4* and memory-optimized *r3*, and three resource types, *large*, *xlarge* and *2xlarge* which posses two, four and eight vCPUs, respectively. Compute-intensive *c4* instances are powered by Intel Xeon E5-2666 v3 processors, *m4* instances are powered by Intel Xeon E5-2676 v3 processors and *r3* instances are powered by Intel Xeon E5-2670 processors. All cloud resources are fully virtualized instances running Ubuntu Linux 16.04 LTS. Maximum of five instances per resource type are allowed in the configuration. All cloud instances are selected from Amazon EC2 Oregon region and the hourly prices range from $0.105 to $0.664.

To apply our modeling approach, we determine cloud resource capacities in terms of instruction execution rate. One way to estimate this rate is to use the base CPU frequency obtained from the specification, and to derive an upper-bound of the performance. However, different applications have different execution profiles and different instruction execution rates, as we show in the previous section. Therefore, to determine the instruction execution rate more accurately, we profile the execution of each application with a small problem size on all resource types. Since cloud virtualization restricts the usage of hardware counters to get instruction count, we execute the application on our local Xeon server to get this value. This local Xeon server and selected cloud resources have the same instruction set architecture (ISA) and the same micro-architecture, hence, the instruction count should be similar. We run the same application on the cloud resources and measure execution time. Measured instruction count is divided by measured execution time to determine

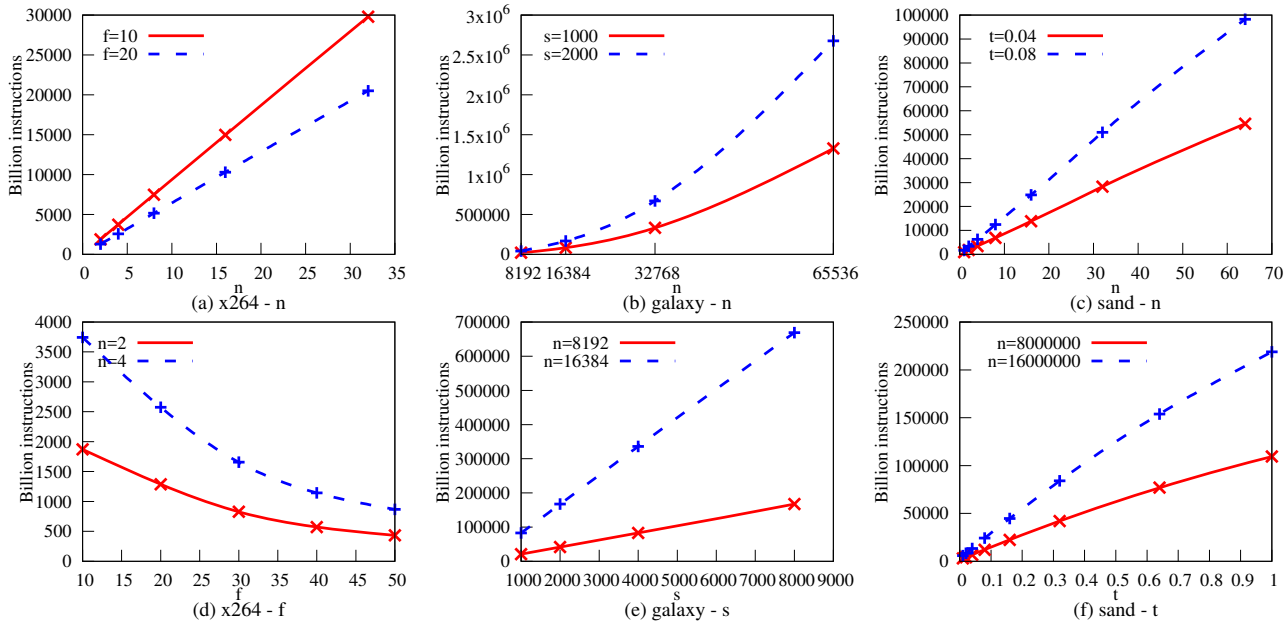


Figure 2. Resource Demand of Elastic Applications

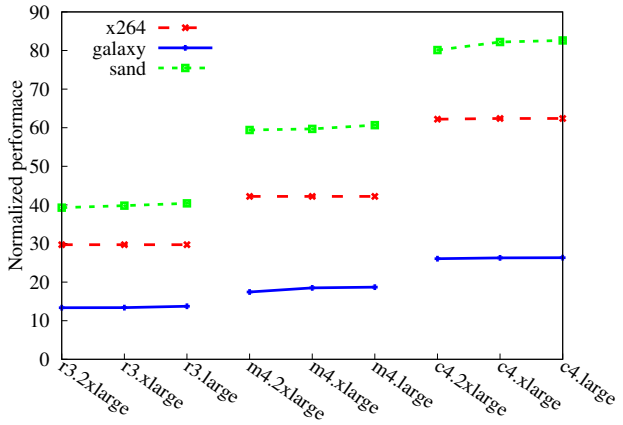


Figure 3. Cloud Resource Characterization

the instruction execution rate for each resource type.

C. Optimizing Resource Characterization

In this section, we show that the user should profile only one resource type for each resource category to get the instruction execution rate per unit cost for one vCPU. Analyzing the ratio between instruction execution rate and cost for each application on all resource types, we observe a large gap between resource categories, as shown in Figure 3. For all applications, *c4* resources have two times better performance per cost compared to *r3* resources, while *m4* resources have *1.5* times better performance per cost compared to *r3* resources. Moreover, resource types within the same resource category have similar performance in terms of instructions executed per unit cost. For example, in galaxy, normalized performances for *c4.large*, *c4.xlarge* and *c4.2xlarge* are 26.27 billion, 26.21 billion and 26.01 billion instructions per second per dollar, respectively. Thus, we safely assume that instruction execution rate per unit cost is similar for all resource types within a resource category. This optimization allows a more practical characterization of a large number of cloud resource types.

D. Validation

To validate our models, we compare the predicted execution time and cost against measured values from Amazon EC2 cloud. Due to research budget constraints, we validate only three cases for each application and show the results in Table IV. Maximum prediction errors are 9.5%, 13.1% and 16.7% respectively for *x264*, *galaxy*, and *sand*. To a large extent, the variation in prediction accuracy could be attributed to the processor sharing approach implemented by cloud providers such as Amazon [26]. The underlying physical processor is shared among multiple cloud instances simultaneously, such that vCPUs are hyper threads and not physical CPU cores. On the other hand, *galaxy* and *sand* have higher prediction error compared to *x264* due to inter-node communication. In contrast, *x264* processes execute standalone on each node with no inter-node communication.

E. Model Analysis

This section presents a model-based analysis of elastic applications of Amazon EC2 cloud resources. This analysis is divided into three parts, covering configuration space, fixed-time scaling and the cost of tightening time deadline. Due to space constraints, we show the results only for galaxy and sand applications, that covers linear, quadratic and logarithmic resource demands.

1) Configuration Space: To understand the implications of large configuration space on cost, we look at the distribution of feasible configurations in the time-cost plane for one problem size. For galaxy, we select a problem size of 65,536 masses with 8,000 simulation steps. For sand, the selected problem size is 8,192 million candidate sequences with 0.32 threshold value. Time deadline is set to 24 hours (hr) and cost budget is set to $350.

We observe a large number of feasible configurations spread-out across the time-cost plane, as shown in Figure 4. There are around 5.8 and 2 million feasible configurations for galaxy and sand, respectively. Moreover, there are multiple Pareto optimal configurations that satisfy both time deadline and cost budget constraints. For galaxy, there are 23 Pareto-optimal configurations spanning covering the cost range $126 - $167. For sand, there are 58 Pareto-optimal

Table IV
MODEL VALIDATION

Application	Configuration	Time (hr)		Cost ($)		Error (%)
		Predicted	Actual	Predicted	Actual	
x264(8000,20)	[2,1,0,0,0,0,0,0]	21	19	23	21	9.5
x264(16000,20)	[5,1,1,0,0,0,0,0]	20	19	46	44	5.0
x264(32000,20)	[5,5,5,1,0,0,0,0]	23	21	99	90	8.7
galaxy(65536,4000)	[5,5,0,0,0,0,0,0]	18	16	53	47	11.1
galaxy(65536,6000)	[5,5,5,0,0,0,0,0]	23	20	85	74	13.0
galaxy(65536,8000)	[5,5,5,3,0,0,0,0]	24	22	126	116	8.3
sand(1024m,0.32)	[5,4,1,0,0,0,0,0]	6	7	18	21	16.7
sand(2048m,0.32)	[5,5,0,0,0,0,0,0]	23	25	72	78	8.6
sand(4096m,0.32)	[5,3,1,0,0,0,0,0]	51	58	144	164	13.7

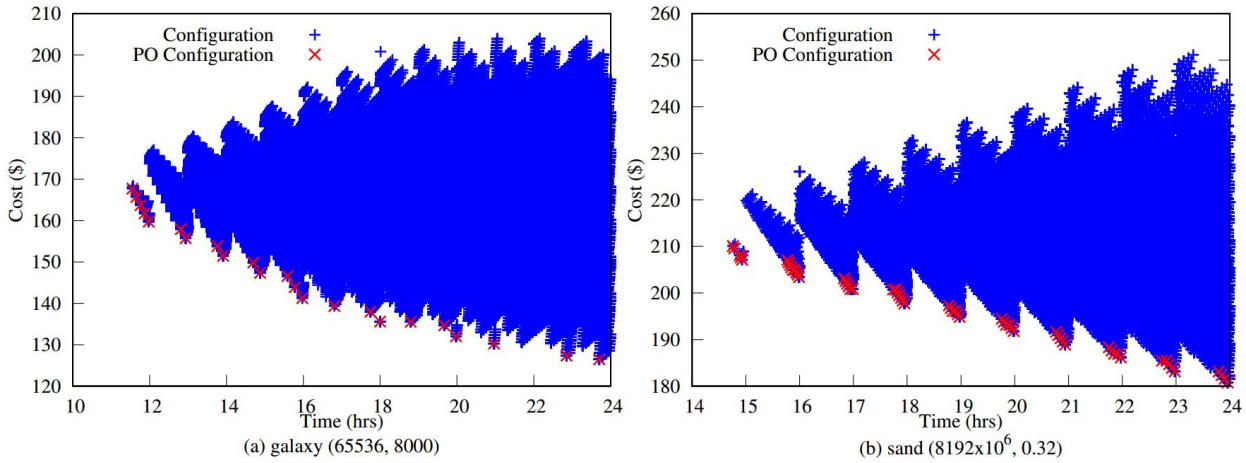


Figure 4. Cloud Resource Configuration Space for 24hr Time Deadline and $350 Cost Budget

configurations with costs spanning from $180 to $210. Among these optimal configurations, there is a considerable span of cost where the highest cost is about 1.3 times and 1.2 times higher compared to the lowest cost, for galaxy and sand, respectively.

Observation 1: Among the large configuration space exposed by cloud resources, there exists a Pareto frontier of multiple Pareto-optimal configurations where cost can be reduced by relaxing the time deadline.

2) Cost of Fixed-Time Scaling: We investigate the implications of fixed-time scaling of elastic applications on cost considering two aspects, (i) scaling problem size and (ii) scaling the accuracy.

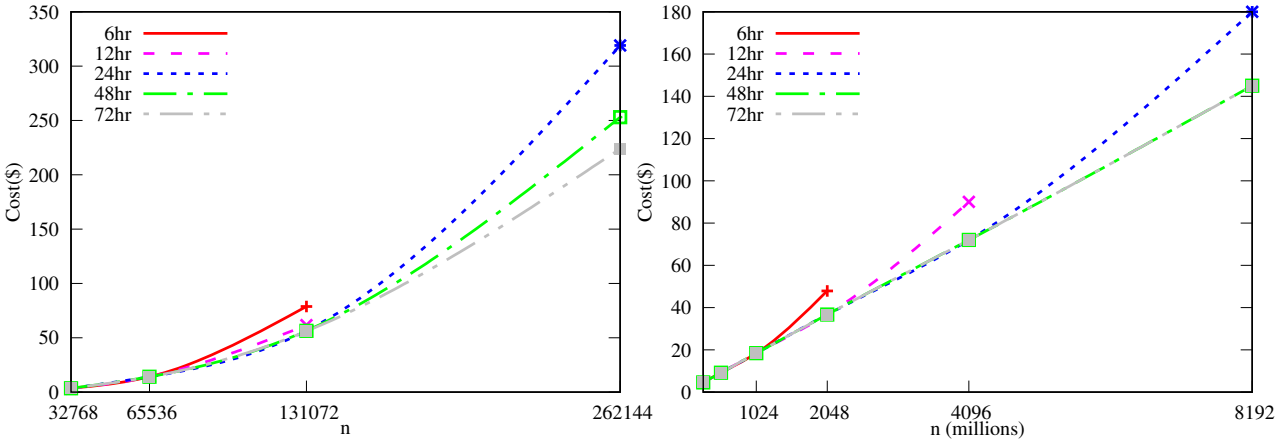
To determine the effects of scaling problem size on cost, we fix the time deadline and accuracy, and determine the minimum cost for different problem sizes. As shown in Section IV-A, the relationship between problem size and resource demand is quadratic for galaxy and linear for sand. As shown in Figure 5, the cost follows approximately the same trend as the resource demand: it grows quadratically with problem size for galaxy and linearly with problem size for sand. However, at some points, curve's gradient is changing and the cost curve deviates from the expected path. We are analyzing this behavior below.

To determine the effects of scaling accuracy on cost, a we fix the time deadline and the problem size, and determine the minimum execution cost for different accuracy. As shown in Figure 6, similar to scaling problem size, growth of cost with scaling accuracy corresponds to the growth of resource demand. For galaxy where the resource demand grows linearly with accuracy, we observe a linear growth of cost while for sand where the resource demand grows logarithmically with accuracy, we observe a logarithmic growth of cost. For applications with sub-linear relationship between cost and accuracy, such as sand, significant accuracy improvements can be achieved for a small increase in cost. As shown in Figure 6(b) for sand, to improve the accuracy of sand 1.6 times, from 0.64 to 1, we need only a 20% increase of the cost.

We observe that cost curves exhibit sudden changes of gradient at certain points when problem size and accuracy is scaled. As shown in Figure 6(a), the cost curve of galaxy for the 24 hours deadline grows linearly till $s = 6,000$ and then it suddenly increases the gradient. To investigate this unexpected behavior, we analyze cloud configurations, as annotated in Figure 6(a)) for each accuracy along the cost curve. In each configuration, first three values represent the numbers of nodes from *c4* category, second three values represent the numbers of nodes from *m4* category and the last three values represent the number of nodes from *r3* category. We observe that whenever the cost curve deviates, the configuration uses a new resource category. In galaxy, when $s = 6,000$ the configuration is [5,5,5,0,0,0,0,0,0], which indicates that maximum number of nodes have been taken from *c4* category. When $s = 8,000$, the configuration [5,5,5,3,0,0,0,0,0] includes additional three nodes from *m4* category. The difference of normalized instruction execution rate across categories, shown in Figure 3, explains this behavior. Since the normalized instruction execution rate of *m4* is higher than the rate of *c4*, the cost gradient increases when the configuration spills into *m4*.

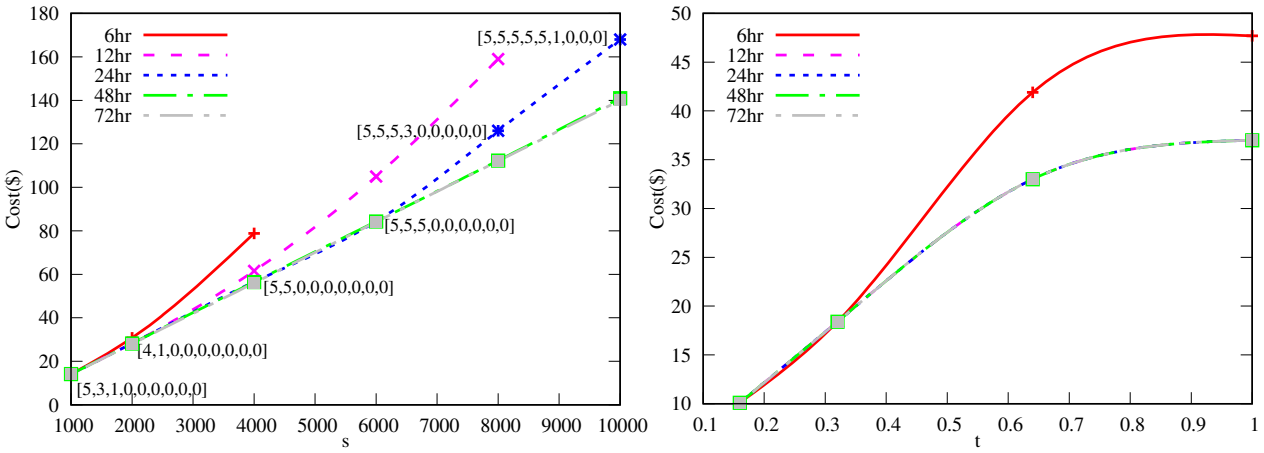
Observation 2: For elastic applications running on cloud, cost grows faster than resource demand when multiple resources having different cost efficiency are used.

3) Cost of Time Deadline: Resource elasticity makes cloud a suitable platform for time critical computations. To understand the impact of time deadline on cost, we fix the problem size and accuracy for galaxy and sand, and observe the variation of minimum cost while tightening time deadline. We observe that the cost increase and the



(a) galaxy - n (b) sand - n

Figure 5. Effect of Scaling Problem Size on Cost



(a) galaxy - s (b) sand - t

Figure 6. Effect of Scaling Accuracy on Cost

time deadline decrease are not proportional. As shown in Figure 5(a) for galaxy(262,144, 1000), tightening the time deadline from 72 hours to 24 hours incurs only $95 of cost increase. Similarly for sand(8192, 0.32), as shown in Figure 5(b), tightening the time deadline from 48 hours to 24 hours incurs only $35 more. Thus, tightening the deadline by 67% results in 40% increase in cost for galaxy, and tightening the deadline by 50% results in 25% increase in cost for sand.

Observation 3: The increase in cost is always smaller than the reduction of execution time when the deadline of an elastic application is tightened.

V. Conclusion

Elastic applications produce results of different accuracy or quality based on the amount of computational resources used. At the same time, clouds are providing a myriad of elastic resource configurations. Thus, choosing the right configuration to execute an application on cloud is a daunting task. This paper introduces CELIA, a measurement-driven analytical modeling approach to determine cloud resource configurations to execute an elastic application with a given time deadline and cost budget. Our execution time and cost models use measured values from baseline runs to estimate application resource demand and cloud resource capacity in terms of executed instructions. We evaluate our approach on Amazon EC2 cloud with a resource configuration space of approximately ten million configurations and three representative elastic applications. Validation against time and cost measurements from Amazon EC2 shows that the prediction error of our models is less than 17%.

We perform a model-based analysis using CELIA and show that multiple cost-time Pareto-optimal configurations

exist for executing elastic applications on the cloud wherein selection of the optimal cloud resource configuration could reduce the execution cost by up to 30%. We investigate the cost of scaling the problem size and accuracy of elastic applications. Our results show that there is potential to trade-off the accuracy of elastic applications for execution cost on cloud. For example, approximately 1.5 times increase of accuracy can be achieved with approximately 1.2 times increase in cost. Moreover, we observe that cost grows faster than application resource demand when multiple resources with different cost efficiency are used. Lastly, we show that the cost of tightening the time deadline is always not proportional to the increase of execution cost. For example, tightening the time deadline by two-thirds increases the cost by just 40%.

VI. Acknowledgements

This work was supported by Singapore Ministry of Education through the Academic Research Fund Tier 1. The authors thank Amazon Web Services for providing Elastic Compute Cloud credits.

References

[1] x264, http://www.videolan.org/developers/x264.html, 2016.

[2] Amazon, AWS Auto Scaling, https://aws.amazon.com/autoscaling, 2016.

[3] Amazon, Amazon EC2 Instance Types, https://aws.amazon.com/ec2/instance-types, 2017.

[4] G. M. Amdahl, Validity of the Single Processor Approach to Achieving Large Scale Computing Capabilities, *Proc. of Spring Joint Computer Conference*, pages 483–485, 1967.

[5] D. Ardagna, G. Casale, M. Ciavotta, J. F. Pérez, W. Wang, Quality-of-Service in Cloud Computing: Modeling Techniques and Their Applications, *Journal of Internet Services and Applications*, 5(1):1, 2014.

[6] T. Bicer, D. Chiu, G. Agrawal, Time and Cost Sensitive Data Intensive Computing on Hybrid Clouds, *Proc. of 12th International Symposium on Cluster, Cloud and Grid Computing*, pages 636–643, 2012.

[7] Y. Gong, A. Zhou, B. He, Monetary Cost Optimizations for MPI-based HPC Applications on Amazon Clouds: Checkpoints and Replicated Execution, *Proc. of International Conference for High Performance Computing, Networking, Storage and Analysis*, page 32, 2015.

[8] Y. Guo, M. Ghanem, R. Han, Does the Cloud Need New Algorithms? An Introduction to Elastic Algorithms, *Proc. of the 4th International Conference on Cloud Computing Technology and Science*, pages 66–73, 2012.

[9] R. Han, Investigations Into Elasticity in Cloud Computing, *arXiv preprint arXiv:1511.04651*, 2015.

[10] J. He, Y. Wen, J. Huang, D. Wu, On the Cost–QoE Tradeoff for Cloud Based Video Streaming Under Amazon EC2's Pricing Models, *IEEE Transactions on Circuits and Systems for Video Technology*, 24(4):669–680, 2014.

[11] H. Huang, L. Wang, B. C. Tak, L. Wang, C. Tang, CAP3: A Cloud Auto-provisioning Framework for Parallel Processing Using On-demand and Spot Instances, *Proc. of 6th International Conference on Cloud Computing*, pages 228–235, 2013.

[12] H. Kllapi, E. Sitaridi, M. M. Tsangaris, Y. Ioannidis, Schedule Optimization for Data Processing Flows on the Cloud, *Proc. of SIGMOD International Conference on Management of Data*, pages 289–300, 2011.

[13] P. Kokkinos, T. A. Varvarigou, A. Kretsis, P. Soumplis, E. A. Varvarigos, Cost and Utilization Optimization of Amazon EC2 Instances, *Proc. of 6th International Conference on Cloud Computing*, pages 518–525, 2013.

[14] S. Leeman-Munk, PetaKit Software Suite: Automated Performance Scaling Analysis, 2010.

[15] Z. Li, H. Zhang, L. OBrien, S. Jiang, Y. Zhou, M. Kihl, R. Ranjan, Spot Pricing in the Cloud Ecosystem: A Comparative Investigation, *Journal of Systems and Software*, 114:1–19, 2016.

[16] T. Lorido-Botran, J. Miguel-Alonso, J. A. Lozano, A Review of Auto-scaling Techniques for Elastic Applications in Cloud Environments, *Journal of Grid Computing*, 12(4):559–592, 2014.

[17] M. Malawski, G. Juve, E. Deelman, J. Nabrzyski, Cost and Deadline Constrained Provisioning for Scientific Workflow Ensembles in IaaS Clouds, *Proc. of International Conference on High Performance Computing, Networking, Storage and Analysis*, page 22, 2012.

[18] M. Mao, M. Humphrey, Auto-scaling to Minimize Cost and Meet Application Deadlines in Cloud Workflows, *Proc. of International Conference for High Performance Computing, Networking, Storage and Analysis*, page 49, 2011.

[19] M. Mao, J. Li, M. Humphrey, Cloud Auto-scaling with Deadline and Budget Constraints, *Proc. of 11th International Conference on Grid Computing*, pages 41–48, 2010.

[20] A. Marathe, R. Harris, D. Lowenthal, B. R. de Supinski, B. Rountree, M. Schulz, Exploiting Redundancy for Cost-effective, Time-constrained Execution of HPC Applications on Amazon EC2, *Proc. of 23rd International Symposium on High-performance Parallel and Distributed Computing*, pages 279–290, 2014.

[21] C. Moretti, A. Thrasher, L. Yu, M. Olson, S. Emrich, D. Thain, PREPRINT: A Framework for Scalable Genome Assembly on Clusters, Clouds, and Grids.

[22] S. Ostermann, A. Iosup, N. Yigitbasi, R. Prodan, T. Fahringer, D. Epema, A Performance Analysis of EC2 Cloud Computing Services for Scientific Computing, *Cloud Computing*, pages 115–131. 2009.

[23] D. Rajan, A. Canino, J. A. Izaguirre, D. Thain, Converting a High Performance Application to an Elastic Cloud Application, *Proc. of 3rd International Conference on Cloud Computing Technology and Science*, pages 383–390, 2011.

[24] U. Sharma, P. Shenoy, S. Sahu, A. Shaikh, A Cost-aware Elasticity Provisioning System for the Cloud, *Proc. of 31st International Conference on Distributed Computing Systems*, pages 559–570, 2011.

[25] S. S. Shende, A. D. Malony, The TAU Parallel Performance System, *International Journal of High Performance Computing Applications*, 20(2):287–311, 2006.

[26] G. Wang, T. E. Ng, The Impact of Virtualization on Network Performance of Amazon EC2 Data Center, *Proc. of IEEE International Conference on Computer Communications*, pages 1–9, 2010.

[27] M. Woodruff, J. Herman, pareto.py: A Epsilon-Nondomination Sorting Routine, https://github.com/matthewjwoodruff/pareto.py, 2013.

[28] A. C. Zhou, B. He, Transformation-based Monetary Cost Optimizations for Workflows in the Cloud, *IEEE Transactions on Cloud Computing*, 2(1):85–98, 2014.